

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-056971

(43)Date of publication of application : 25.02.2000

(51)Int.Cl.

G06F 9/38
G06F 9/45

(21)Application number : 10-220329

(71)Applicant : FUJITSU LTD

(22)Date of filing : 04.08.1998

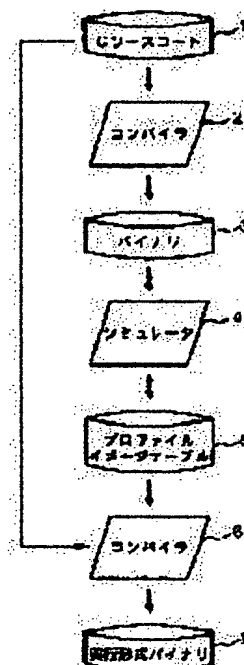
(72)Inventor : IWATA YASUSHI

(54) HIGH-SPEED ARITHMETIC PROCESSOR AND RECORDING MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To execute a program in data dependency relation fast by extracting information on data dependency relation and performing static prediction when a source program is compiled, and dynamically predicting and selectively executing an instruction which is not solved so far and has high prediction probability at the time of execution.

SOLUTION: A compiler 2 takes a morpheme analysis and a syntax analysis of a C source code 1 to generate a binary code. A simulator 4 registers in a profile image table 5 a pair of the line of an instruction in data dependency relation of an object to be predicted and a dependent instruction among instructions in data dependency relation. In the profile image table 5, the object instruction to be predicted among the instructions in the data dependency relation is registered. A compiler 6 generates a binary code 7 in execution form and registers an instruction code and a dependent instruction in the data dependence relation in the binary code in execution form corresponding to line numbers (address). The analysis and prediction of the source program and the fast execution of the compiled instruction in executable form are enabled.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2000 Japan Patent Office

おねナシ

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号
特開2000-56971
(P2000-56971A)

(43)公開日 平成12年2月25日(2000.2.25)

| | | | |
|--------------------------|-------|--------------|-------------------|
| (51)Int.Cl. ⁷ | 識別記号 | F I | テマコード*(参考) |
| G 0 6 F 9/38 | 3 5 0 | G 0 6 F 9/38 | 3 5 0 A 5 B 0 1 3 |
| 9/45 | | 9/44 | 3 2 2 F 5 B 0 8 1 |

審査請求 未請求 請求項の数8 O L (全 9 頁)

| | | | |
|----------|---------------------|----------|---|
| (21)出願番号 | 特願平10-220329 | (71)出願人 | 000005223 富士通株式会社 神奈川県川崎市中原区上小田中4丁目1番1号 |
| (22)出願日 | 平成10年8月4日(1998.8.4) | (72)発明者 | 岩田 靖 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内 |
| | | (74)代理人 | 100089141 弁理士 岡田 守弘 |
| | | Fターム(参考) | 5B013 AA00 AA12 CC00 5B081 AA06 CC11 CC21 |

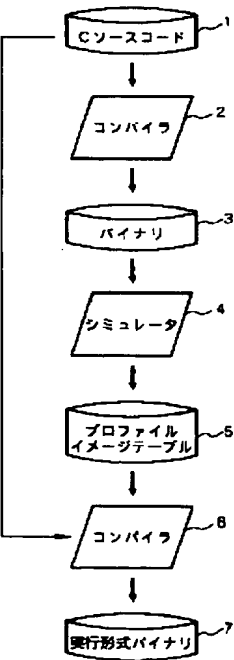
(54)【発明の名称】 高速演算処理装置および記録媒体

(57)【要約】

【課題】 本発明は、ソースプログラムを解析して予測を行う高速演算処理装置および記録媒体に関し、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までには解決されなくて予測確率の高い命令を動的予測して選択実行し、データ依存関係にあるプログラムの高速実行を実現することを目的とする。

【解決手段】 ソースプログラムを解析する手段と、解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをONに設定する手段とを備えるように構成する。

本発明のシステム構成図



【特許請求の範囲】

【請求項1】 ソースプログラムを解析して予測を行う高速演算処理装置において、
ソースプログラムを解析する手段と、
上記解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、
上記設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをONに設定する手段とを備えたことを特徴とする高速演算処理装置。

【請求項2】 上記予測フラグをONに設定する命令として、自タスク内でデータが決定される命令を除いた命令としたことを特徴とする請求項1記載の高速演算処理装置。

【請求項3】 上記予測フラグをONに設定する命令として、他タスク内でデータが決定されるが命令実行時までには確定している命令を除いた命令としたことを特徴とする請求項1あるいは請求項2記載の高速演算処理装置。

【請求項4】 ソースプログラムをコンパイルした実行可能形式の命令を高速実行する高速演算処理装置において、
実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行する手段と、
上記動的予測の結果の集計を行い予測の正解率の低い命令の先行実行を抑止する手段とを備えたことを特徴とする高速演算処理装置。

【請求項5】 上記動的予測の結果の集計として、予測結果が正解のときに正解フラグに+1、不正解のときに不正解フラグを+1して正解率を集計することを特徴とする請求項4記載の高速演算処理装置。

【請求項6】 上記データ依存関係にある命令として、メモリアクセス命令としたことを特徴とする請求項1から請求項5のいずれかに記載の高速演算処理装置。

【請求項7】 ソースプログラムを解析する手段と、
上記解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、
上記設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをONに設定する手段ととして機能させるプログラムを記録したコンピュータ読取可能な記録媒体。

【請求項8】 実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行する手段と、
上記動的予測の結果の集計を行い予測の正解率の低い命令の先行実行を抑止する手段ととして機能させるプログラムを記録したコンピュータ読取可能な記録媒体。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、ソースプログラムを解析して予測を行うおよびソースプログラムをコンパ

イルした実行可能形式の命令を高速実行する高速演算処理装置および記録媒体に関するものである。

【0002】

【従来の技術】 従来、プロセッサの実行性能向上を図るために分岐命令があるときに分岐先を予測して先行的に実行し、予測結果が正しければその先行実行した次から実行を継続し、予測結果が間違っていれば先行実行をキャンセルして他の分岐先を実行するようにしていた。

【0003】

【発明が解決しようとする課題】 上述したように従来は分岐先の予測、即ち制御の流れの分岐予測は行われていたが、あるタスク内で実行に必要なデータが他のタスク内で実行されて確定しないと実行できないといういわゆるデータ依存関係がある場合のデータ予測は行われておらず、高速実行し得ないという問題があった。

【0004】 本発明は、これらの問題を解決するため、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までには解決されなくて予測確率の高い命令を動的予測して選択実行し、データ依存関係にあるプログラムの高速実行を実現することを目的としている。

【0005】

【課題を解決するための手段】 図1を参照して課題を解決するための手段を説明する。図1において、コンパイラ2は、Cソースコードを形態素解析、構文解析などを行い、バイナリを生成するものである。

【0006】 シミュレータ4は、データ依存関係にある命令のうち予測すべき命令を選択してプロファイルイメージテーブル5に登録するものである。プロファイルイメージテーブル5は、データ依存関係にある命令のうち予測すべき対象の命令に登録するものである（後述する図4参照）。

【0007】 実行形式バイナリ7は、実行形式のバイナリ（後述する図5参照）である。次に、動作を説明する。コンパイラ2がCソースプログラムを解析し、シミュレータ4が解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をプロファイルイメージテーブル5に設定および設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをONに設定するようにしている。

【0008】 この際、予測フラグをONに設定する命令として、自タスク内でデータが決定される命令を除いた命令とするようにしている。また、予測フラグをONに設定する命令として、他タスク内でデータが決定されるが命令実行時までには確定している命令を除いた命令とするようにしている。

【0009】 また、実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行し、動的予測の結果の集

計を行い予測の正解率の低い命令の先行実行を抑止するようにしている。

【0010】この際、動的予測の結果の集計として、予測結果が正解のときに正解フラグに+1、不正解のときに不正解フラグを+1して正解率を集計するようにしている。

【0011】また、データ依存関係にある命令として、メモリアクセス命令とするようにしている。従って、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までに解決されなくて予測確率の高い命令を動的予測して選択実行することにより、データ依存関係にあるプログラムを高速実行することが可能となる。

【0012】

【発明の実施の形態】次に、図1から図13を用いて本発明の実施の形態および動作を順次詳細に説明する。

【0013】図1は、本発明のシステム構成図を示す。図1において、Cソースコードは、ソースコードの例であって、ここでは、C言語のソースコードである。

【0014】コンパイラ2は、Cソースコード1を形態素解析、構文解析などを行い、バイナリ3を生成するものである。バイナリ3は、コンパイラ2によって生成された実行形式のコードである。

【0015】シミュレータ4は、データ依存関係にある命令のうち予測すべき命令を選択してプロファイルイメージテーブル5に登録などするものである。プロファイルイメージテーブル5は、データ依存関係にある命令のうち予測すべき対象の命令に登録するものであって、後述する図4に示すようにデータ依存関係にある命令の行番号、依存命令の対に登録したものである。

【0016】コンパイラ6は、実行形式バイナリ7を生成するものである。実行形式バイナリ7は、実行形式のバイナリであって、後述する図5に示すように行番号

(アドレス)に対応づけて命令コード、データ依存関係にある依存命令に登録したものである。以下順次詳細に説明する。

【0017】図2は、本発明の説明図(その1)を示す。図2において、PE(A)、PE(B)は、プロセッサエレメントであって、タスクA、Bを動作させるものである。ここでは、PE(A)上でタスクAが動作し、PE(B)上でタスクBが動作している。一点鎖線の矢印でデータ依存関係(1)、(2)、(3)を示し、このうちのデータ依存関係(1)のみが、データ予測して先行実行する対象である。

【0018】PC(行番号)は、プログラムカウンタであって、現在の実行しようとする行番号(アドレス)を保持(生成)するものである。次に、図2の動作を説明する。

【0019】(1) PE(B)上で動作するタスクB

が行番号PC=001で $z = y + 4$ の演算を行う。この際、演算中のyはPE(A)で動作するタスクAが行番号PC=101で $y = c + 2$ の演算によって算出されて確定するまで待機する必要があるというデータ依存関係がある。

【0020】(2) PE(B)上で動作するタスクBが行番号PC=002で $z = z + 2$ の演算を行う。この際、同じタスクBで行番号PC=001で既に算出されて確定しており待機する必要があるというデータ依存関係がある。

【0021】(3) PE(B)で動作するタスクBが行番号PC=003で $x = x + 5$ の演算を行う。この際、演算中のxはPE(A)で動作するタスクAが行番号PC=100で $x = a + 1$ の演算によって算出されて確定しており待機する必要があるというデータ依存関係がある。

【0022】従って、上記(1)、(2)、(3)のデータ依存関係のうち、待機する必要があるデータ依存関係は(1)のみでありこれが本願発明のデータの値を予測して先行実行する対象の命令であり、後述する図5の実行形式バイナリの該当するアドレス(行番号)の予測フラグを1(予測)に設定する。予測する必要のないものには予測フラグを0(非予測)に設定する。これにより、コンパイル時にデータ依存関係にある命令のうち静的な予測が行われたこととなる。以上のことをまとめると以下ようになる。

【0023】(イ) 自タスク内で定義されている命令は予測対象としない(図2の(2)の場合は予測対象としない)。

(ロ) 他タスク内で定義されていて、その命令を実行するまでにそのデータが確定している命令は予測対象としない(図2の(3)の場合は予測対象としない)。

【0024】(ハ) 他タスク内で定義されていて、その命令を実行するときでも、そのデータが確定しない命令は予測対象とする(図2の(1)の場合は予測対象とする)。

【0025】(ニ) 尚、後述する図8で説明するように、(ハ)の命令についてデータを予測して実際にシミュレーション実行して予測結果の正解率が高い命令のみを予測対象とし、正確率が低い命令を予測対象から外すようにしてもよい。

【0026】図3は、本発明の説明図(その2)を示す。これは、図2のタスクB上で動作するソースイメージプログラムの例を示す。ここでは、例えば図2のタスクB上で実行される行番号001は、001 add %rz, %ry, 4となる。001は行番号であり、addは加算命令であり、%rz, %ry, 4はレジスタryの内容(y)に4を加算してその結果をレジスタrzに格納するものである。同様に、行番号121、200は、ST(ストア命令)、ld(ロード命令)につい

て記述したものである。

【0027】図4は、本発明のプロファイルイメージテーブル例を示す。これは、データ依存関係にある命令を
プログラムカウンタ 依存命令1
001 101

は、図2のタスクBで実行される行番号001の命令であって、データ依存関係の1番目が行番号101である旨を表す。データ依存関係の2番目、3番目はここではなしである。行番号200の命令は、データ依存関係のある命令が1番目（依存命令1）150（レジスタr2）、2番目（依存命令2）130（レジスタr3）、3番目（依存命令3）121（レジスタr8）の3つがある。

| アドレス | 命令コード | 依存命令1 | 依存命令2 | 依存命令3 | 予測フラグ |
|------|---------------|-------|-------|-------|--------|
| 001 | add %rz,%ry,4 | 101 | — | — | 1（予測） |
| 002 | add %rz,%rz,z | — | — | — | 0（非予測） |
| 003 | add %rx,%rx,5 | 100 | — | — | 0（非予測） |

以上のようにデータ依存関係にあるアドレス（行番号）、命令コード、依存命令1～3、予測フラグfをコンパイル時に登録しておくことにより、実行時に予測フラグが1（予測）のみの命令についてデータの値を予測して先行実行することにより、選択的に静的予測した結果をもとに効率的に実行時にデータ依存関係にある命令を先行実行し、CPUの高速実行を実現することが可能となる。

【0030】図6は、本発明の他のシステム構成図を示す。これは、実行時に後述する図8に示すように、PC（プログラムカウンタ）に設定されたアドレス（行番号）とタグとを比較して一致するときにヒット、一致するものみつからないときにミスヒットを出力し、ヒットのときは更に値と差分を読み出して演算器によって演算（ここでは、例えば加算）して予測値を算出するハードイメージである。

【0031】以上の構成により、図5で予測フラグが1（予測）とした命令のアドレスとそのときの変数のデータの値、および前前回と前回との差分を高速に読み出し、データ依存関係にある変数の値を予測して先行実行することが可能となる。以下実行時の動作を順次詳細に説明する。

【0032】図7は、本発明の説明図（その3、実行時）を示す。これは、PC（プログラムカウンタ）に行番号001が設定され、図2のPE（B）上で動作するタスクBが行番号001で $z = y + 4$ を実行しようとしたときの様子を示す。

【0033】初期値は、データ依存関係にある行番号001の命令の変数yの初期値が0であることを表す。1回目は、データ依存関係にある行番号001の命令の変数yの実行結果が8となったことを表す。

【0034】2回目は、データ依存関係にある行番号001の命令の変数yの予測の例を表す。（a）は、2回

ソースプログラムから抽出して登録したものであって、図2の行番号001、002、003のものを登録したものである。例えば

依存命令2 依存命令3
— —

【0028】以上のように、コンパイル時に命令の行番号に対応づけてデータ依存関係のある他の命令の行番号を抽出して登録することにより、データ依存関係にある命令についてのみデータの値を予測して先行実行することが可能となる。

【0029】図5は、本発明の実行形式バイナリ例を示す。これは、実行形式のバイナリであって、図示の下記の項目を登録したものである。

目に予測した値 $y = 16$ （1回目の値に差分を加算した $8 + 8 = 16$ ）と実際に実行したyの値16とが一致して正解の場合を表す。この場合には、図示の下記のように更新する。

【0035】

| PC | 値 | 差分 |
|-----|----|----|
| 001 | 16 | 8 |

（b）は、2回目に予測した値 $y = 16$ （1回目の値に差分を加算した $8 + 8 = 16$ ）と実際に実行したyの値8とが不一致で不正解の場合を表す。この場合には、図示の下記のように更新する。

【0036】

| PC | 値 | 差分 |
|-----|---|----|
| 001 | 8 | 0 |

以上のように、初期値をもとに1回目の値を予測して実際の実行結果と一致したときにそのときの値および前回との差分を登録することを繰り返すと、後述する図8に示すようなテーブルを作成することが可能となる。そして、正解のときに予測した実行結果が正しいのでそれに続く処理を実行することが可能となり、一方、不正解のときに予測の結果をキャンセルして確定した変数の値で再実行する。これにより、実行時に正解となる値および差分が図8のテーブルに示すように登録され、当該正解数の割合が所定閾値（例えば60%以上）の命令の予測を行い、それ以外の命令の予測を抑止することで、予測率の確率を向上させてキャンセル率を小さくして全体の実行速度を大幅に向上させることが可能となる。

【0037】図8は、本発明の説明図（その4、実行時）を示す。これは、既述した図7の手順によって予測対象の命令（命令の行番号）について正解のときの変数の値および前回との差分を登録したリストであり、正解と不正解の数を集計して正解率が所定閾値（例えば60%以上）の命令のみを残し他の命令の予測フラグを0

(非予測)に設定し、動的に統計を集計してデータ依存関係にあるデータの予測の正解率を向上させて高速実行させることを動的に制御することが可能となる。ここでは、 $PC=001$ (行番号001)について n 回実行し、正解が13974回、不正解が64回となり、正解率が60%以上であるので、データの依存関係にある命令(行番号001の命令)のデータ予測を行う。一方、 $PC=002$ (行番号002)について n 回実行し、正解が21回、不正解が17365回となり、正解率が60%以下であるので、データの依存関係にある命令(行番号002の命令)のデータ予測を抑止し、キャンセルによる不要な負荷による処理速度の低下を防止することを実行時に動的に行うことが可能となる。

【0038】図9は、本発明の1実施例構成図を示す。図9の(a)は、実行オブジェクトの各PE#1から#4への割り振りの様子を示す。ここでは、図示のように、実行オブジェクトを4台のPE#1から#4に割り振ると共に、このときに各アドレス(行番号)に対応づけて既述したデータ依存関係にある命令のアドレス(行番号)を依存命令1、2、3に登録しておく(コンパイル時に収集して登録しておく)。これら4台のPE#1から#4にそれぞれ割り振られた実行オブジェクトは、逐次(b)のPE#1から#4によって実行する。

【0039】図9の(b)は、4台のPE#1から#4によってそれぞれ実行オブジェクトを実行するときの処理を示す。ここでは、各PE#1から#4では、データ依存ありがYES、依存解決済みがNOの場合には、変数の値を予測し、この予測した値をもとに実行オブジェクトをそれぞれ実行する。

【0040】以上のように、実行オブジェクトを各PE#1から#4に割り振り、各PE#1から#4でデータ依存ありがYES、依存解決済みがNOの場合に、変数の値を予測し、この予測した値をもとに実行オブジェクトをそれぞれ実行することにより、データ依存関係にある命令を正解率高く先行実行し、処理速度を高速化することが可能となる。

【0041】図10は、本発明の他の動作説明フローチャートを示す。これは、既述したデータ依存関係にある命令の変数の値の予測を動的に決定するときのフローチャートである。

【0042】図10において、S1は、依存ありか判別する。これは、例えば既述した図9の(a)で各PEに割り振られた実行オブジェクトに依存命令1、2、3のいずれかが設定され、実行しようとする命令にデータ依存関係があるか判別する。YESの場合には、データ依存関係があると判明したので、S2に進む。NOの場合には、データ依存関係がないと判明したので、S5で予測フラグを0(非予測)にセットする。

【0043】S2は、データ依存関係があったが解消したか判別する。YESの場合には、データ依存関係が解

消したのでS6で予測フラグを0(非予測)にセットする。NOの場合には、データ依存関係が解消していないので、S3に進む。

【0044】S3は、変数の値の予測した結果の正解率が高いか判別する。これは、命令の変数の値の予測を行ってその結果を既述した図8に示すように収集してその正解率が高い(例えば60%以上)のときにデータ予測して高速化できると判明したので、S4で命令の変数のデータの値を予測する。一方、NOの場合には、正解率が低く、キャンセルに伴う処理が発生し、予測を行うと処理速度がそのために低下してしまうので、S7で予測フラグを0(非予測)にセットする。

【0045】以上のように、実行時に動的にデータの依存関係があり、データの依存関係が解消していなく、かつデータ予測したときの正解率が高いときに命令の変数のデータの値の予測を行い、それ以外の命令についてデータの予測を抑止し、命令の実行速度を動的に解析して効率的に高速化を図ることが可能となる。

【0046】図11は、本発明の予測正解率例を示す。これは、ベンチマークプログラムSC(SPECint92, System Performance Evaluation Cooperative)をシミュレート実行したときのデータであって、横軸が各種処理を行うプログラムなどを表し、縦軸が予測正解率を表す。図中の棒グラフの横線の部分がLOAD命令の部分を表し、横線の部分がSTORE命令の部分を表し、点々の部分がOTHERS(LAAD、STOREなどのメモリアクセス命令以外の命令)の部分を表す。図中の記号は下記を表す。

【0047】

comp : compress (圧縮プログラム)

eqn : eqntott (Cプログラム)

esp : esprese

gcc : コンパイラ

li : lispインタプリタで9クイーン問題を解く

sc : スプレッドシート

ave : 平均

図12は、本発明の選択的値予測の対象命令と非対象命令例(%)を表す。これは、図11で既述したベンチマークプログラムSCを分類したものであって、図示の下記の項目に分類したものである。

【0048】

・依存無し(データ依存関係がない場合) :

・解決済み(データ依存関係はあるが実行時まで解決済みの場合) :

・予測対象(データ依存関係があり、解決済みでない場合) : 本願発明の命令のデータの値の予測対象となる命令

・load : データをメモリにロードする命令

・store : メモリからデータをレジスタにストアす

る命令

・others: load、storeのメモリアクセス命令以外の命令

ここで、左側に記載した(1)は、図2の(1)に対応して、データ依存関係を予測する対象の割合(%)である。

【0049】(2)は、図2の(2)に対応して、データ依存関係がない命令の割合(%)である。(3)は、図2の(3)に対応して、データ依存関係があるか命令の実際の実行時までには解決してデータの値が確定している命令の割合(%)である。

【0050】従って、ベンチマークプログラムSCでは、予測対象は、comp、eqn、esp、gcc、li、sc、Aveで9.2~11.6%の間の割合であり、これらについて選択的に命令の変数の値の予測を行い、むやみに予測する際のキャンセルに伴う処理の低下を無くし、予測したときに高速化される可能性の高いもののみのデータ依存関係にある命令の変数の値を予測して先行実行することが可能となる。更に、load命令やstore命令のメモリアクセスは実行速度が遅く先行実行するメリットが高いため、当該メモリアクセス命令(load命令、store命令)を予測対象として選択し、データ依存関係にある命令の変数の値を予測して先行実行するようにしてもよい。

【0051】図13は、本発明の実行時間例を示す。これは、既述したベンチマークプログラムSCの各プログラムをシミュレート実行したときの時間を表し、横軸が各種処理を行うプログラムなどを表し、縦軸が実行時間(Kサイクル)を表す。図中の棒グラフの左側が予測なし、中央が予測あり、右側が完全予測(予測が全部正解の場合)を表す。例えばscの場合には、予測なしの場合に比して、本願発明の予測を行うと、20.9%の時間(サイクル数)だけ高速化できることが判明した。詳細に説明すれば、ベンチマークプログラムSC中に含まれる9.2%のsc(スプレッドシート)の命令を予測対象とし、約20.9%のサイクル数だけ高速化できることが判明した。この際、予測を全て正解とすると、約

34.5%のサイクル数だけ高速化できるとシミュレートできた。ベンチマークプログラム中の他のプログラムについても図12に示す全体に対するそれぞれの割合に対し、図13の中央の予測ありのサイクル数に高速化できる。

【0052】

【発明の効果】以上説明したように、本発明によれば、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までには解決されなくて予測確率の高い命令を動的予測して選択実行する構成を採用しているため、データ依存関係にあるプログラムを高速実行する装置を実現できる。

【図面の簡単な説明】

【図1】本発明のシステム構成図である。

【図2】本発明の説明図(その1)である。

【図3】本発明の説明図(その2)である。

【図4】本発明のプロファイルイメージテーブル例である。

【図5】本発明の実行形式バイナリ例である。

【図6】本発明の他のシステム構成図である。

【図7】本発明の説明図(その3、実行時)である。

【図8】本発明の説明図(その4、実行時)である。

【図9】本発明の1実施例構成図である。

【図10】本発明の他の動作説明フローチャートである。

【図11】本発明の予測正解率例である。

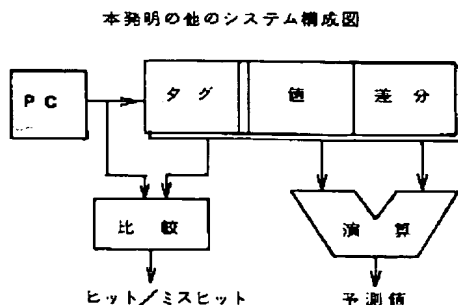
【図12】本発明の選択的値予測の対象命令と非対象命令例である。

【図13】本発明の実行時間例である。

【符号の説明】

- 1: ソースコード
- 2、6: コンパイラ
- 3: バイナリ
- 4: シミュレータ
- 5: プロファイルイメージテーブル
- 7: 実行形式バイナリ

【図6】



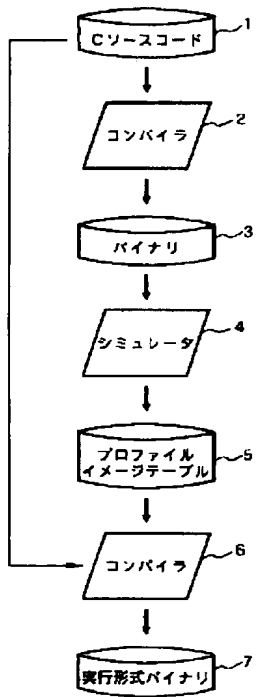
【図12】

本発明の選択的値予測の対象命令と非対象命令例[%]

| | comp | eqn | esp | gcc | li | sc | Ave |
|----------|------|------|------|------|------|------|------|
| (2) 依存無し | 52.4 | 47.3 | 52.0 | 57.8 | 53.0 | 48.2 | 51.8 |
| (3) 解決済 | 38.1 | 42.0 | 38.9 | 32.0 | 35.4 | 42.6 | 38.1 |
| (1) 予測対象 | 9.5 | 10.7 | 9.1 | 10.2 | 11.6 | 9.2 | 10.2 |
| load | 15.8 | 14.5 | 24.5 | 22.0 | 42.6 | 18.9 | 23.0 |
| store | 31.8 | 14.7 | 19.7 | 14.7 | 13.7 | 13.5 | 18.0 |
| others | 52.3 | 70.8 | 55.8 | 63.4 | 43.6 | 67.6 | 58.9 |

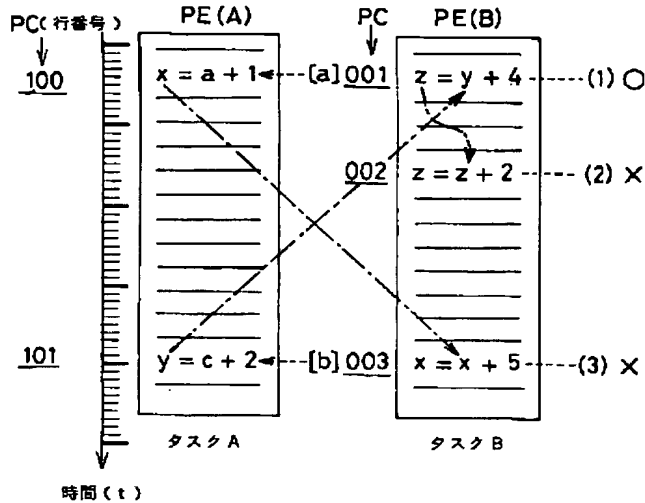
【図1】

本発明のシステム構成図



【図2】

本発明の説明図（その1）



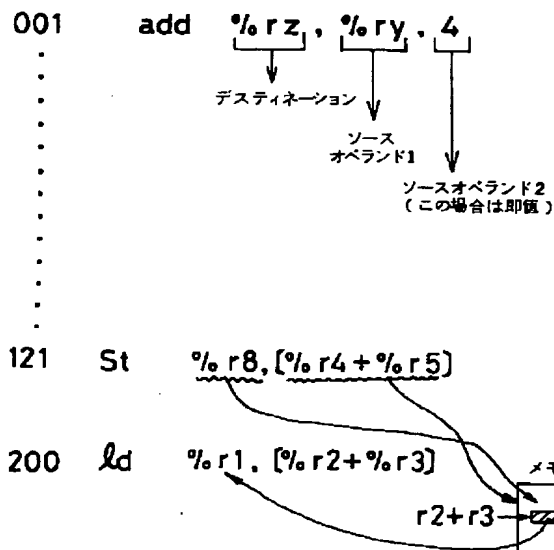
【図4】

本発明のプロファイルイメージテーブル例

| プログラム カウンタ | 依存 命令 1 | 依存 命令 2 | 依存 命令 3 |
|---------------|------------|------------|------------|
| 001 | 101 | --- | --- |
| 002 | --- | --- | --- |
| 003 | 100 | --- | --- |
| 200 | 150(r2) | 130(r3) | 121(r8) |

【図3】

本発明の説明図（その2）

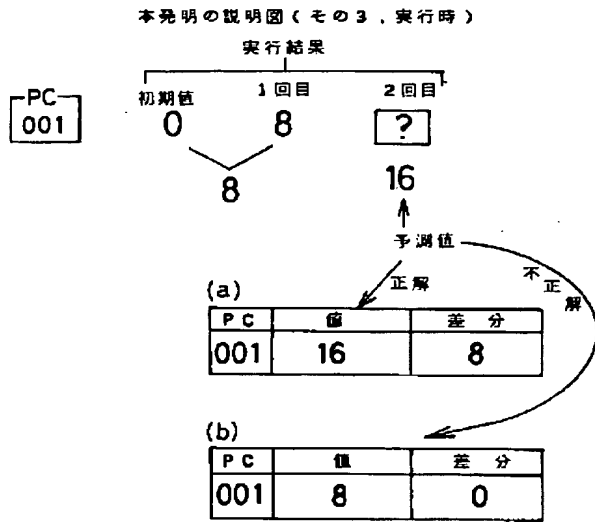


【図5】

本発明の実行形式バイナリ例

| アドレス | 命令コード | 依存 命令 1 | 依存 命令 2 | 依存 命令 3 | 予測フラグ |
|------|-------------------|------------|------------|------------|-------|
| 001 | add %rz, %ry, 4 | 101 | --- | --- | 1 |
| 002 | add %rz, %rz, 2 | --- | --- | --- | 0 |
| 003 | add %rz, %rz, 5 | 100 | --- | --- | 0 |
| . | . | . | . | . | . |
| 200 | ld %r1, [%r2+%r3] | 150(r2) | 130(r3) | 121(r8) | . |

【図 7】



【図 10】

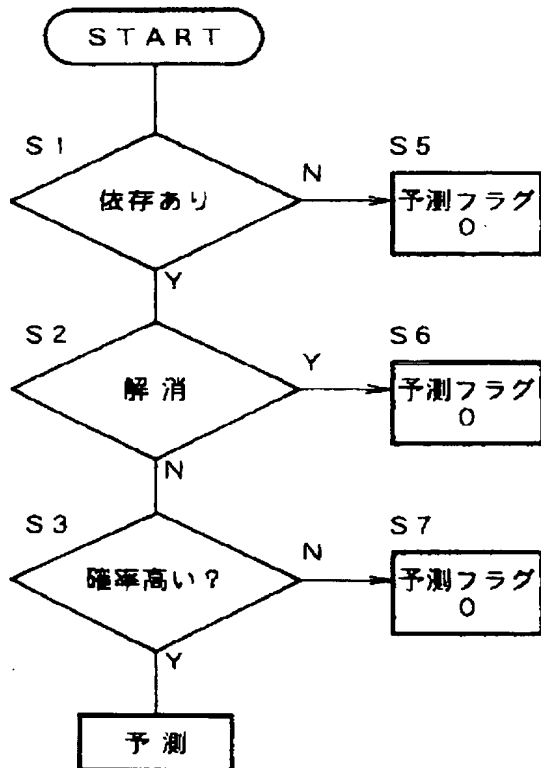
【図 8】

本発明の説明図（その 4、実行時）

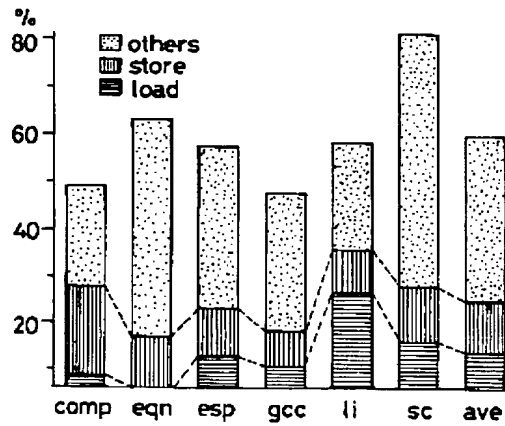
| | PC | 値 | 差 | 正解 | 不正解 |
|-------------|-----|----|---|-------|-------|
| 1 回目 | 001 | 0 | 0 | 0 | 0 |
| 2 回目 不正解 | 001 | 8 | 8 | 0 | 1 |
| 3 回目 正解 | 001 | 16 | 8 | 1 | 1 |
| 4 回目 不正解 | 001 | 16 | 0 | 1 | 2 |
| | . | . | . | . | . |
| | . | . | . | . | . |
| n 回 | 001 | 32 | 2 | 13974 | 64 |
| | . | . | . | . | . |
| n 回 | 002 | 64 | 4 | 21 | 17385 |

【図 11】

本発明の他の動作説明フローチャート

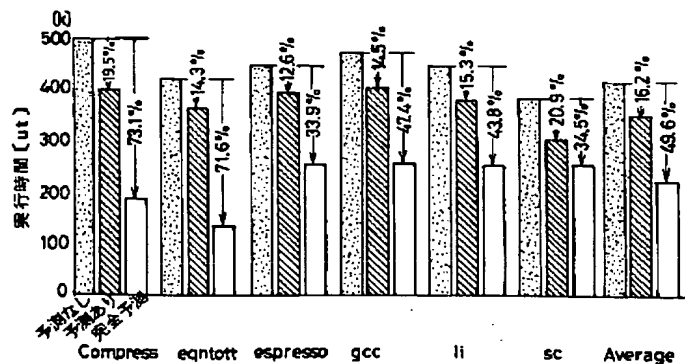


本発明の予測正解率例



【図 13】

本発明の実行時間例



本発明の１実施例構成図

